AI Communications (2013). 26(4), 355-371.
DOI: http://dx.doi.org/10.3233/AIC-130573

A statistical simulation technique to develop and evaluate conversational agents

David Griol, Javier Carbo and José M. Molina
Applied Artificial Intelligence Group (GIAA), Department of Computer Science, Carlos III University of Madrid, Avda. de la Universidad, 30, 28911, Leganés, Spain
E-mails: {david.griol, javier.carbo, josemanuel.molina}@uc3m.es

Abstract: In this paper, we present a technique for developing user simulators which are able to interact and evaluate conversa-tional agents. Our technique is based on a statistical model that is automatically learned from a dialog corpus. This model is used by the user simulator to provide the next answer taking into account the complete history of the interaction. The main objective of our proposal is not only to evaluate the conversational agent, but also to improve this agent by employing the simulated dialogs to learn a better dialog model. We have applied this technique to design and evaluate a conversational agent which provides academic information in a multi-agent system. The results of the evaluation show that the proposed user simulation methodology can be used not only to evaluate conversational agents but also to explore new enhanced dialog strategies, thereby allowing the conversational agent to reduce the time needed to complete the dialogs and automatically detect new valid paths to achieve each of the required objectives defined for the task.

Keywords: Conversational agents, user simulation, statistical methodologies, multiagent systems

1. Introduction

The widespread use of new mobile technology implementing wireless communications enables a new type of advanced applications to access information on the Internet. Speech and natural language technologies allow users to communicate in a flexible and efficient manner, making possible to access these applications where traditional input interfaces cannot be used (e.g. in-car applications, access for disabled persons, etc). Also speech-based interfaces work seamlessly with small devices and allow users to easily invoke local applications or access remote information. For this reason, conversational agents [14,19,34,37,46] are becoming a strong alternative to traditional graphical interfaces which might not be appropriate for all users and/or applications. These agents can be defined as software application that accepts natural language as input and generates natural language as output, engaging in a conversation with the user. To successfully manage the interaction with the users, conversational agents usually carry out five main tasks: automatic speech recognition (ASR), natural language understanding (NLU), dialog management (DM), natural language generation (NLG) and text-to-speech synthesis (TTS).

There is a high variety of applications in which conversational agents can be used, one of the most widespread of which is information retrieval. Some sample applications are tourist and travel information [23], weather forecast over the phone [65], speech controlled telephone banking systems [38], conference help [7], etc. Spoken interaction can be the only way to access information in some cases, like for example when the screen is too small to display information (e.g. handheld devices) or when the eyes of the user are busy in other tasks (e.g. driving) [59]. It is also useful for remote control of devices and robots, specially in smart environments [39]. One of the most recent applications of these agents is for the development of e-learning and tutoring systems [8,33,58]. Finally, embodied conversational agents and companions [10,22] are currently one of the most demanding applications for fully natural and understandable dialogs.

One of the core aspects of developing conversational agents is to design flexible dialog management strategies. The dialog strategy defines the behavior of the conversational agent in response to user utterances and environmental states that, for example, can be based on observed or inferred events or beliefs. This is the fun-

damental task of dialog management [44], as the performance of the system depends to a great extent on the quality of this strategy. However, there is no clear definition of what constitutes a good strategy [64], and thus a great effort is employed to design them and find empirical evidence of their appropriateness. This design is usually carried out in industry by hand-crafting dialog strategies tightly coupled to the application domain in order to optimize the behavior of the conversational agent in that context. However, it is a very time-consuming process and has the disadvantage of lack of portability and adaptation to new contexts.

This has motivated the research community to find ways for automating dialog learning by using statistical models trained with real conversations. Statistical approaches can model the variability in user behaviors and allow exploring a wider range of strategies. Although the construction and parameterization of the model depends on expert knowledge of the task, the final objective is to develop conversational agents that have a more robust behavior, better portability, and are easier to adapt to different user profiles or tasks. As the success of statistical approaches to model the system depends on the quality of the data used to develop the statistical dialog model, considerable effort is necessary to acquire and label a corpus with the data necessary to train a good model. In order to mitigate this, user simulators appeared as an efficient means to generate dialogs between the system and a simulated user [53]. This way, the user simulator makes it possible to generate a large number of dialogs in a very simple way, therefore reducing the time and effort required for the evaluation of a conversational agent, as well as allowing to evaluate it in early development phases.

In this paper, we present a technique to develop a user agent simulator to automatically interact with a conversational agent and generate the dialogs required to learn an enhanced dialog models. Our user simulation technique is based on a classification process that takes into account the previous dialog history to select the next user answer. We have applied this technique to develop a conversational agent which provides academic information in Spanish. The results of the evaluation of the conversational agent show that the conversational agent reduces the time needed to fulfill the different tasks, thereby allowing the conversational agent to manage new dialog situations and generate better answers for the situations already present in an initial dialog model.

The remainder of the paper is as follows. Section 2 reviews different approaches related to the simulation of multiagent systems. This section focuses on the description of statistical techniques for user simulation and conversational agents. Section 3 describes our proposal to develop a user simulator based on a statistical model that is learned from a dialog corpus. Section 4 presents a detailed explanation of how our proposal has been applied to develop a practical conversational agent that works as an academic assistant. Sections 5 and 6 respectively detail the process and criteria that we have employed to evaluate our proposal and discuss the evaluation results. Finally, our conclusions and future work are presented in Section 7.

2. Related work

This section describes the application of user simulation techniques in multiagent systems (MAS). Firstly, the main applications and proposals of these techniques within the framework of MAS are presented. Secondly, the section is focused on the application of user modeling techniques within the fields of language processing and conversational agents. A comparison between the most representative proposals within these fields and the main contributions of our proposed technique to develop a user simulator are then introduced.

The term computer simulation is defined in [5] as "*the usage of a computational model to gain additional insight into a complex system's behavior (e.g. biological or social systems) by envisioning the implications of the modeling choices, but also to evaluate designs and plans without actually bringing them into existence in the real world*". Agent-based simulation (ABS) is a relatively recent modeling technique widely used to model these complex systems with applications in many disciplines ranging from logistics optimization [60], biological systems [4], traffic conditions [3], pedestrian simulation [2], urban planning and simulation [42], social sciences [45] and economics [63]. Detailed studies on agent-based modeling and simulation can be found in [5,28,36].

Despite this extreme heterogeneity of simulated application domains, the different approaches often share the common viewpoint on the modeled system represented by an individual agent acting and interacting with other entities in a shared environment. Thus, the main elements in simulation models are agents with a possibly heterogeneous behavior, the environment that provides perceptions and enables their actions, and mechanisms of interaction among agents involving the exchange of information and the effects of the percep-

tions and corresponding actions decided by the different agents. ABS models address different issues, for instance, the system has still not fully completed, ethical reasons (e.g., the safety of humans would be involved), practical reasons (e.g., reduce the time and costs that are required to develop and evaluate the system), etc.

Considering the growing interest of simplifying the construction and evaluation of MAS, it is not surprising the number of software frameworks specifically aimed at supporting the design and implementation of agent-based simulation systems. This kind of frameworks often provides not only abstractions and mechanisms for the definition of agents and their environments, but also additional functionalities for the management of the simulation, its visualization, monitoring and the acquisition of data about the simulated dynamics. A first category of these platforms provides general purpose frameworks in which agents mainly represent passive abstractions interacting in an overall simulation process (e.g., NetLogo [61]). A second category of platforms are based on general purpose programming languages providing very similar support tools (e.g., Repast [43]). A third category of platforms represents an attempt to provide a higher level linguistic support, trying to reduce the distance between agent-based models and their implementations (e.g., SimSesam [30]).

2.1. User modeling and natural language processing

Research in techniques for user modeling has a long history within the fields of language processing and conversational agents. The main purpose of a simulated user in this field is to improve the usability of a conversational agent through the generation of corpora of interactions between the system and simulated users [41], reducing time and effort required for collecting large samples of interactions with real users. Moreover, each time changes are made to the system it is necessary to collect more data in order to evaluate the changes. Thus, the availability of large corpora acquired with a user simulator should contribute positively to the development of the system.

User simulators can be used to evaluate different aspects of a conversational agent, particularly at the earlier stages of development, or to determine the effects of changes to the system's functionalities (e.g., evaluate confirmation strategies or introduce of errors or unpredicted answers in order to evaluate the capacity of the dialog manager to react to unexpected situations). A second usage, in which we are mainly interested in this contribution, is to support the automatic learning of optimal dialog strategies using statistical methodologies. Large amounts of data are required for a systematic exploration of the dialog state space and corpora acquired with simulated users are extremely valuable for this purpose.

Two main approaches can be distinguished to the creation of simulated users: rule based and data or corpus based. In a rule-based simulated user the researcher can create different rules that determine the behavior of the system [12,32,35]. This approach is particularly useful when the purpose of the research is to evaluate the effects of different dialog management strategies. In this way the researcher has complete control over the design of the evaluation study.

An alternative approach, often described as corpus-based or data-based, uses probabilistic methods to generate the user input, with the advantage that this uncertainty can better reflect the unexpected behaviors of users interacting with the system. Statistical models for modeling user behavior have been suggested as the solution to the lack of the data that is required for training and evaluating dialog strategies. Using this approach, the dialog manager can explore the space of possible dialog situations and learn new potentially better strategies. Methodologies based on learning user intentions have the purpose of optimizing dialog strategies. A summary of user simulation techniques for reinforcement learning of the dialog strategy can be found in [53].

The most extended methodology for machine-learning of dialog strategies consists of modeling human-computer interaction as an optimization problem using Markov Decision Process (MDP) and reinforcement methods [31]. The main drawback of this approach is the large state space of practical spoken dialog systems, whose representation is intractable if represented directly. Although Partially Observable MDPs (POMDPs) outperform MDP-based dialog strategies, they are limited to small-scale problems, since the state space would be huge and exact POMDP optimization is again intractable [62].

In [15,16], Eckert, Levin and Pieraccini introduced the use of statistical models to predict the next user action by means of a n-gram model. The proposed model has the advantage of being both statistical and task-independent. Its weak point consists of approximating the complete history of the dialog by a bigram model. In [31], the bigram model is modified by considering only a set of possible user answers following a given system action (the Levin model). Both models have the

drawback of considering that every user response depends only on the previous system turn. Therefore, the simulated user can change objectives continuously or repeat information previously provided.

In [54,55], Scheffler and Young propose a graph-based model. The arcs of the network symbolize actions, and each node represents user decisions (*choice points*). In-depth knowledge of the task and great manual effort are necessary for the specification of all possible dialog paths.

Pietquin, Beaufort and Dutoit combine characteristics of the Scheffler and Young model and Levin model. The main objective is to reduce the manual effort necessary for the construction of the networks [47]. A Bayesian network is suggested for user modeling. All model parameters are hand-selected.

Georgila, Henderson and Lemon propose the use of HMMs, defining a more detailed description of the states and considering an extended representation of the history of the dialog [21]. Dialog is described as a sequence of *Information States* [9]. Two different methodologies are described to select the next user action given a history of information states. The first method uses n-grams [15], but with values of n from 2 to 5 to consider a longer history of the dialog. The best results are obtained with 4-grams. The second methodology is based on the use of a linear combination of 290 characteristics to calculate the probability of every action for a specific state.

Cuayáhuitl et al. present a method for dialog simulation based on HMMs in which both user and system behaviors are simulated [13]. Instead of training only a generic HMM model to simulate any type of dialog, the dialogs of an initial corpus are grouped according to the different objectives. A submodel is trained for each one of the objectives, and a bigram model is used to predict the sequence of objectives.

In [50], a new technique for user simulation based on explicit representations of the user goal and the user agenda is presented. The user agenda is a structure that contains the pending user dialog acts that are needed to elicit the information specified in the goal. This model formalizes human–machine dialogs at a semantic level as a sequence of states and dialog acts. An EM-based algorithm is used to estimate optimal parameter values iteratively. In [52], the agenda-based simulator is used to train a statistical POMDP-based dialog manager.

A data-driven user intention simulation method that integrates diverse user discourse knowledge (cooperative, corrective, and self-directing) is presented in [29]. User intention is modeled based on logistic regression and Markov logic framework. Human dialog knowledge is designed into two layers, domain and discourse knowledge, and integrated with the data-driven model in generation time. A methodology of user simulation applied to the evaluation and refinement of stochastic dialog systems is presented in [57]. The proposed user simulator incorporates several knowledge sources, combining statistical and heuristic information to enhance the dialog models by an automatic strategy learning. As it is described in the following section, our proposed user simulation technique is based on a classification process that considers the complete dialog history by incorporating several knowledge sources, combining statistical and heuristic information to enhance dialog models by an automatic strategy learning.

3. Our proposal to develop a user agent simulator

Our proposed architecture to provide context-aware services by means of conversational agents is described in [27]. It consists of five different types of agents that cooperate to provide an adapted service. *User agents* are configured into mobile devices or PDAs. *Provider Agents* supply the different services in the system and are bound to *Conversational Agents* that provide the specific services. A *Facilitator Agent* links the different positions to the providers and services defined in the system. A *Positioning Agent* communicates with the ARUBA positioning system to extract and transmit positioning information to other agents in the system [49]. Finally, a *Log Analyzer Agent* generates user profiles that are used by Conversational Agents to adapt their behavior taking into account the preferences detected in the users' previous dialogs. Figure 1 shows the complete architecture designed for the overall MAS, in which a real user is replaced by the proposed user simulation technique.

The interaction between the different agents in the architecture is as follows. The ARUBA positioning system is used to extract information about the positions of the different agents in the system. This way, it is possible to know the positions of the different User Agents and thus extract information about the Conversational Agents that are available in the current location. The Positioning Agent reads the information about position provided by the ARUBA system and communicates this information to the User Agent. Once a User Agent is aware of its own location, it communicates this information to the Facilitator Agent in order to find out the different services available in that

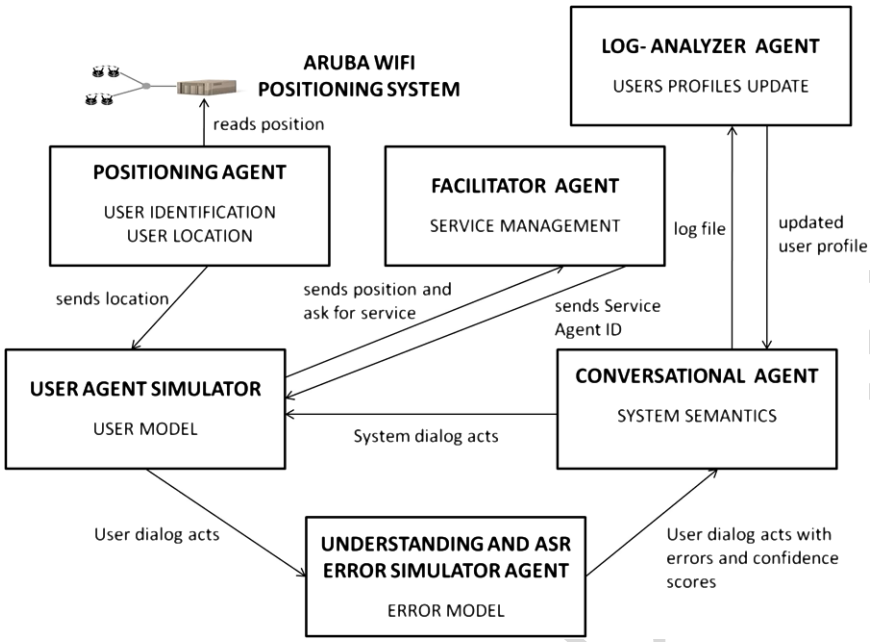

Fig. 1. Graphical scheme of the complete agent-based architecture for the user simulation technique. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AIC-130573.)

location. The Facilitator Agent informs the User Agent about the services available in this position.

Once the User Agent has selected a specific service, it communicates its decision to the Facilitator Agent and queries it about the service providers that are available. The Facilitator Agent informs the User Agent about the identifier of the Conversational Agent that supplies the required service in the current location. Then, the User Agent asks the Conversational Agent for the required service and the conversational agent manages the dialog providing the specific service. Once the interaction with the Conversational Agent has finished, the Conversational Agent reads the contents of the log file for the dialog and send this information to the Log Analyzer Agent. The Log Analyzer Agent stores this log file and generates a user profile to personalize future services. This profile is sent to the Conversational Agent at the beginning of each new interaction.

3.1. The user agent simulator

The user simulator that we present in this paper replaces the user agent in our proposed architecture. This agent simulates the user intention level, that is, the simulator provides concepts and attributes that represent the intention of the user utterance. Therefore, the user simulator carries out the functions of the ASR and NLU modules, i.e., it generates the semantic interpretation of the user utterance in the same format defined for the output of the NLU module.

The methodology that we have developed for user simulation extends our work for developing a statistical methodology for dialog management [26]. The user answers are generated taking into account the information provided by the simulator throughout the history of the dialog, the last system turn, and the objective(s) predefined for the dialog.

In order to control the interaction, our user simulator uses the representation the dialogs as a sequence of pairs (A_i, U_i), where A_i is the output of the dialog system (the system answer) at time i, expressed in terms of dialog acts; and U_i is the semantic representation of the user turn (the result of the understanding process of the user input) at time i, also expressed in terms of dialog acts This way, each dialog is represented by $(A_1, U_1), \ldots, (A_i, U_i), \ldots, (A_n, U_n)$, where A_1 is the greeting turn of the system (the first turn of the dialog), and U_n is the last user turn. We refer to a pair (A_i, U_i) as S_i, the state of the dialog sequence at time i.

In this framework, we consider that, at time i, the objective of the user simulator is to find an appropriate user answer U_i. This selection is a local process for each time i and takes into account the sequence of dialog states that precede time i, the system answer at time i, and the objective of the dialog $\mathcal{O}$. If the most

probable user answer U_i is selected at each time i, the selection is made using the maximization:

$$\hat{U}_i = \arg\max_{U_i \in \mathcal{U}} P(U_i|S_1, \ldots, S_{i-1}, A_i, \mathcal{O}),$$

where set $\mathcal{U}$ contains all the possible user answers.

As the number of possible sequences of states is very large, we establish a partition in this space (i.e., in the history of the dialog preceding time i).This data structure, that we call *User Register* (*UR*), contains the information provided by the user throughout the previous history of the dialog. After applying the above considerations and establishing the equivalence relations in the histories of the dialogs, the selection of the best U_i is given by:

$$\hat{U}_i = \arg\max_{U_i \in \mathcal{U}} P(U_i|UR_{i-1}, A_i, \mathcal{O}).$$

As in our previous work on dialog management [25], we propose the use of a multilayer perceptron (MLP) [6,48] to make the determination of the next user turn. The input layer receives the current situation of the dialog, which is represented by the term $(UR_{i-1}, A_i, \mathcal{O})$ in the previous equation. The values of the output layer can be viewed as the a posteriori probability of selecting the different user answers defined for the simulator given the current situation of the dialog. The choice of the most probable user answer of this probability distribution leads to the previous equation. In this case, the user simulator will always generate the same response for the same situation of the dialog. Since we want to provide the user simulator with a richer variability of behaviors, we base our choice on the probability distribution supplied by the MLP on all the feasible user answers, not only selecting the most probable user response for each dialog situation.

For the conversational agent to determine the next response, we have assumed that the exact values provided by the user simulator are not significant. They are important for accessing the databases and for constructing the output sentences of the conversational agent. However, the only information necessary to determine the next action by this agent is the presence or absence of specific information. Therefore, the information we used from the *UR* is a codification of this data in terms of three values, $\{0, 1, 2\}$, for each field in the *UR* according to the following criteria:

- **0**: The value of the specific position of the *UR* has not been provided by the user simulator.
- **1**: The value of the specific position of the *UR* has been provided with a confidence score that is higher than a given threshold. Confidence scores are provided by the Understanding and ASR Error Simulator Agent, as it is explained in Section 3.2.
- **2**: The value of the specific position of the *UR* has been provided with a confidence score that is lower than the given threshold.

3.2. The understanding and ASR error simulator agent

A real corpus includes information about the errors that were introduced by the ASR and the NLU modules during the acquisition. This information also includes confidence measures, which are used by the conversational agent to evaluate the reliability of the concepts and attributes generated by the NLU module. This way, an error simulator agent has been designed to perform error generation. This agent modifies the dialog acts generated by the user simulator once the *UR* is updated. In addition, the error simulator adds confidence scores to the semantic representation generated by the user simulator.

One of the main problems that must be considered during the interaction with a conversational agent is the propagation of errors through the different modules in the system. The recognition module must deal with the effects of spontaneous speech and with noisy environments; consequently, the sentence provided by this module could incorporate some errors. The understanding module could also add its own errors (which are mainly due to the lack of coverage of the semantic domain). Finally, the semantic representation provided to the dialog manager might also contain certain errors. Therefore, it is desirable to provide the dialog manager with information about what parts of the user utterance have been clearly recognized and understood and what parts have not.

In our proposal, the user simulator provides the conversational agent with the dialog act representation associated to the user input together with its confidence scores [20]. To do this, an error simulation agent has also been incorporated in the architecture described in [27] to include semantic errors in the generation of dialogs. This agent modifies the dialog acts provided by the user agent simulator once it has selected the information to be provided to the user. In addition, the error simulation agent adds a confidence score to each concept and attribute in the semantic representation generated for each user turn.

For the study presented in this paper, we have improved this agent using a model for introducing errors based on the method presented in [51]. The generation of confidence scores is carried out separately from the model employed for error generation. This model is represented as a communication channel by means of a generative probabilistic model $P(c, a_u|\tilde{a}_u)$, where a_u is the true incoming user dialog act $\tilde{a}_u$ is the recognized hypothesis, and c is the confidence score associated with this hypothesis.

The probability $P(\tilde{a}_u|a_u)$ is obtained by Maximum-Likelihood using the initial labeled corpus acquired with real users and considers the recognized sequence of words w_u and the actual sequence uttered by the user $\tilde{w}_u$. This probability is decomposed into a component that generates a word-level utterance from a given user dialog act, a model that simulates ASR confusions (learned from the reference transcriptions and the ASR outputs), and a component that models the semantic decoding process:

$$P(\tilde{a}_u|a_u) = \sum_{\tilde{w}_u} P(a_u|\tilde{w}_u) \times \sum_{w_u} P(\tilde{w}_u|w_u)P(w_u|a_u).$$

Confidence score generation is carried out by approximating $P(c|\tilde{a}_u, a_u)$ assuming that there are two distributions for c. These two distributions are hand-crafted, generating confidence scores for correct and incorrect hypotheses by sampling from the distributions found in the training data corresponding to our initial corpus:

$$P(c|a_u, \tilde{a}_u) = \begin{cases} P_{\text{corr}}(c) & \text{if } \tilde{a}_u = a_u, \\ P_{\text{incorr}}(c) & \text{if } \tilde{a}_u \neq a_u. \end{cases}$$

The dialog manager of the conversational agent considers that a dialog is not successful when one of the following conditions takes place: (i) the dialog exceeds a maximum number of system turns, usually higher than the average number of turns of the dialogs acquired with real users; (ii) the answer selected by the dialog manager corresponds to a query not made by the user simulator; (iii) the database query module generates an error because the user simulator has not provided the mandatory data needed to carry out the query; (iv) the answer generator generates an error when the selected answer involves the use of a data item not provided by the user simulator. A user request for closing the dialog is selected once the system has provided the information defined in its objective(s). The dialogs that fulfill this condition before the maximum number of turns are considered successful.

4. Design of an academic conversational agent

The design of our conversational agent is based on the requirements defined for the UAH dialog system [11], which was developed to provide spoken access to academic information about the Department of Languages and Computer Systems in the University of Granada. To successfully manage the interaction with the users, the conversational agent carry out six main tasks described in the Introduction section: automatic speech recognition, natural language understanding, dialog management, database access and storage, natural language generation, and text-to-speech synthesis.

The dialog manager for the conversational agent has been developed using a technique that combines the VoiceXML standard [56] with a statistical dialog manager [24]. VoiceXML files and grammars are simplified by generating a VoiceXML document for each specific system prompt, as can be observed in the left corner of Fig. 2. Each file contains a reference to a grammar that defines the valid user's inputs for the corresponding system prompt, as observed at the right corner of the figure. The conversational agent selects the next system prompt (i.e. VoiceXML file) by verifying the probabilities assigned by the statistical dialog manager to each system prompt given the current state of the dialog. The interaction of the user simulator with the conversational agent allows the modification of these probabilities each time a new successful dialog is simulated.

The information that the conversational agent provides has been classified into four groups: subjects, professors, doctoral studies and registration. The information that the agent provides for each of these categories is shown in Table 1. As can be observed, the system must ask the user for different pieces of information before producing a response. The way in which the user is queried for this information follows in most cases a system-directed initiative.

As in many other conversational agents, the semantic representation that we have chosen for the task is based on the concept of frame [40]. Frames are a way of representing semantic knowledge. A frame is a structure for representing a concept or situation. Each concept in a domain has usually associated a group of attributes (slots) and values [18]. In this semantic representation, one or more concepts represent the intention of the utterance, and a sequence of attribute-value

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml xmlns="www.w3.org/2001/vxml"
   xmlns:xsi="www.w3.org/2001/
   XMLSchema-instance"
   xsi:schemaLocation="www.w3.org/2001/vxml
   www.w3.org/TR/voicexml20/vxml.xsd"
   version="2.0" application="app-UAH.vxml">
<form id="subject_form">
 <field name="subject_name">
    <grammar type="application/srgs+xml"
          src="/grammars/subjects.grxml"/>
    <prompt>Tell me the name of the subject.
    </prompt>
    <filled>
      <return namelist="subject_name"/>
    </filled>
  </field>
 </form>
</vxml>
```

```
#JSGF V1.0;
grammar subjects;
public <subject_name> = [<desire>]
[<information>] {this.subjects=$subject_name}]
[<connector> {this.subjects=$degree}]
<desire> = I want [to know] | I would like
[to know] | I would like | I want | I need
| To know about;
<information> = information about the subject |
information about | the subject;
<subject_name> = Language Processors |
Compilers | Computers Basics | Physics for
Computer Science | Computer Technology |
Statistics | Databases | Operating Systems I |
Operating Systems II | Formal Languages and
Automata Theory | Requirements Engineering |
Computer Networks | Artificial Intelligence |
Logic Design | ... ;
<connector> = of the | of | ... ;
<degree> = Computer Science | Biotechnology |
Telecommunication | ... ;
```

Fig. 2. VoiceXML document to prompt the user for the name of a subject (*left*) and grammar to recognize this specific information (*right*).

Table 1
Information provided by the academic conversational agent

Category	Information provided by the user (names and examples)		Information provided by the system
Subject	*Name*	Compilers	Degree, lecturers, responsible lecturer, semester, credits, web page
	Degree, in which it is taught in case that there are several subjects with the same name	Computer Science	
	Group name and optionally *type*, in case the user asks for information about a specific group	A Theory A	Timetable, lecturer
Lecturers	Any combination of *name* and *surnames*	John John Smith Mr. Smith	Office location, contact information (phone, fax, email), groups and subjects, doctoral courses
	Optionally *semester*, in case the user asks for the tutoring hours	First semester Second semester	Tutoring hours
Doctoral studies	Name of a doctoral program	Software development	Department, professor
	Name of a course if the user asks for information about a specific course	Object-Oriented Programming	Type, credits
Registration	Name of the deadline	Provisional registration confirmation	Initial time, final time, description

pairs contains the information about the values given by the user.

In the case of user turns, we defined four concepts related to the different queries that the user can perform to the system (*Subject*, *Lecturers*, *Doctoral studies*, *Registration*), three task-independent concepts (*Affirmation*, *Negation*, and *Not-Understood*), and eight attributes (*Subject-Name*, *Degree*, *Group-Name*, *Subject-Type*, *Lecturer-Name*, *Program-Name*, *Semester* and *Deadline*). An example of the semantic interpretation of a user's sentence is shown below:

User turn:
I want to know information about the subject Language Processors I of Computer Science.

Semantic representation:
(*Subject*)
Subject-Name: Language Processors I
Degree: Computer Science

The labeling of the system turns is similar to the labeling defined for the user turns. A total of 30 task-dependent concepts was defined:

- Task-independent concepts (*Affirmation*, *Negation*, *Not-Understood*, *New-Query*, *Opening*, and *Closing*).
- Concepts used to inform the user about the result of a specific query (*Subject*, *Lecturers*, *Doctoral-Studies*, and *Registration*).
- Concepts defined to require the user the attributes that are necessary for a specific query (*Subject-Name*, *Degree*, *Group-Name*, *Subject-Type*, *Lecturer-Name*, *Program-Name*, *Semester*, and *Deadline*).
- Concepts used for the confirmation of concepts (*Confirmation-Subject*, *Confirmation-Lecturers*, *Confirmation-DoctoralStudies*, *Confirmation-Registration*) and attributes (*Confirmation-SubjectName*, *Confirmation-Degree*, *Confirmation-SubjectType*, *Confirmation-LecturerName*, *Confirmation-Semester*, *Confirmation-Program-Name*, *Confirmation-Deadline*, and *Confirmation-GroupName*).

An example of the semantic interpretation of a system prompt is shown below:

System turn:
Do you want to know the name of the lecturer of the Group 1 of Language Processors I?

Semantic representation:
(*Lecturers*)
Group-Name: 1
Subject-Name: Language Processors I

The User Register (*UR*) defined for the UAH task is a sequence of 12 fields, corresponding to the four concepts (*Subject*, *Lecturers*, *Doctoral-Studies*, and *Registration*) and eight attributes (*Subject-Name*, *Degree*, *Group-Name*, *Subject-Type*, *Lecturer-Name*, *Program-Name*, *Semester*, and *Deadline*) defined for the task. This information is coded as explained at the end of Section 3.1.

To train and evaluate the neural networks for the user simulator designed for the UAH task, we used the *April* toolkit, developed by the Technical University of Valencia [17]. *April* is an efficient implementation of the Backpropagation (BP) algorithm to train neural networks with general feedforward topology. In addition, *April* adds a *matrix* and a *dataset* class which allow the straightforward definition and manipulation of huge sets of samples more flexibly than simply enumerating the pairs of inputs and outputs.

In order to successfully use neural networks as classifiers, a number of considerations had to be taken into account, such as the network topology, the training algorithm, and the selection of the parameters of the algorithm. The number of input units in the MLP for the user simulator was set by the size of the parameters introduced in the codification of the last system prompt, the current user register, and the objective defined for the dialog. There was one output unit corresponding to each different user response defined in the corpus. Using *April*, topology and algorithm parameters (i.e. learning rate and momentum) are estimated with an exhaustive search, using as stop criteria the MSE obtained in each epoch for the validation set. The gradient is computed in incremental mode, this way the weights are updated after each input, also a momentum is added to the backpropagation so that the networks can overcome local minimums.

Different experiments were completed using different network topologies of increasing number of weights: a hidden layer with 2 units, two hidden layers of 2 units each, two hidden layers of 4 and 2 units, a hidden layer with 4 units, etc. Several learning algorithms were also tested: the incremental version of the backpropagation algorithm (with and without momentum term) and the quickprop algorithm. The influence of their parameters such as learning rate or momentum term was also studied. We firstly tested the influence of the topology of the MLP, by training different MLPs of increasing number of weights using the standard backpropagation algorithm (with a sigmoid activation function and a learning rate equal to 0.2), and selecting the best topology according to the mean square error (MSE) of the validation data. The minimum MSE of the validation data was achieved using an MLP of one hidden layer of 32 units. We followed our experimentation with MLPs of this topology, training MLPs with several algorithms: the incremental version of the backpropagation algorithm (with and without momentum term) and the quickprop algorithm. The best result on the validation data was obtained using the MLP trained with the standard backpropagation algorithm and a value of LR equal to 0.3.

5. Experiments

The process that we have employed to evaluate our proposal is summarized in Fig. 3. Firstly, several ex-

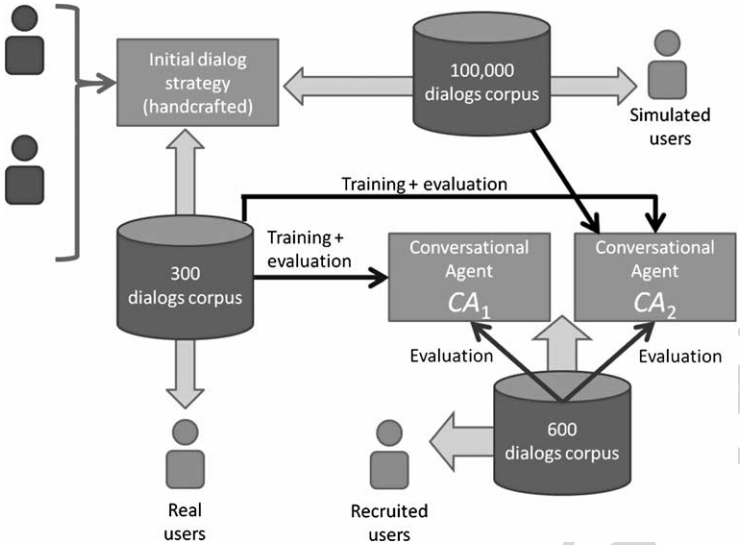

Fig. 3. Experimental set-up.

perts developed an initial, handcrafted dialog strategy for the UAH system, resembling the behavior of experienced human operators who provide the same information that the system [11]. After implementing this strategy, a corpus of 300 dialogs was collected from spontaneous telephone calls to the system, which contains 1,853 user utterances. Figure 4 shows a sample dialog (S = system, U = user). A conversational agent called CA_1 was trained with this corpus.

Secondly, we employed the simulation technique described in Section 3 and generated 100,000 dialogs. The goal was to explore the space of possible dialog states to enable the DM learning dialog strategies that are better than the initial strategy created by the experts. To do so, a set of 33 different scenarios was created for the user simulator, with the same goals as those of the real users in the corpus. To check the evolution in the dialog strategy, we employed a second dialog manager, called CA_2, which was trained using the successful simulated dialogs and the dialog corpus collected from the real users.

5-fold cross-validation was used to evaluate CA_1 and CA_2. The corpus for each conversational agent was randomly split into five folds, each containing 20% of the corpus. The experiments were carried out in five trials, each using as a test set a different fold whereas the remaining folds were used as the training set. A validation subset (20%) was extracted from each training set. We carried out a detailed study of the dialogs obtained with both conversational agents using the set of quantitative evaluation measures proposed in [1], as will be discussed in Section 6.

S1: Welcome to the university information system. You can consult information about subjects, lecturers, doctoral studies and registrations.
U1: I want to know information about subjects in Computer Science.
S2: Which subject do you want to consult?
U2: Language Processors I.
S3: Here you can consult the information of the subject. Anything else?
U3: Can you inform me about the schedules of Group D?
S4: I give the information for the Group D of Language Processors I: Monday 9:00-11:00, Wednesday 11:00-13:00. Anything else?
U4: Can you tell me the name of the lecturer?
S5: I give the information for the Group D of Language Processors I. The name of the lecturer is David Griol. Anything else?
U5: No, thank you.
S6: Thank you for using the system.

Fig. 4. Example of a dialog acquired with real users (translation from Spanish to English).

Figure 5 shows an example of a simulated dialog from the acquired corpus with two objectives: to know timetables of a given lecturer and subject, and obtain information about the registration in its specific degree. The confidence scores generated by the error simulator are shown between brackets. Sentences in natural language, that are equivalent to the labeled system and user turns, have been included to clarify the explanation.

S1: (Opening)
Welcome to the UAH system. You can consult information about subjects, lecturers, doctoral studies and registrations.
U1: Lecturers [0.2]
I want to know information about lecturers.
S2: (Confirmation:Lecturers)
Do you want to know information about lecturers?
U2: Affirmation [0.9]
Yes
S3: (Question:Lecturer_Name)
Tell me the name of the lecturer.
U3: Lecturer_Name: John Smith [0.1]
John Smith
S4: (Confirmation:Lecturer_Name)
Do you want to know the information about John Smith?
U4: Affirmation [0.8]
Yes
S5: (Answer:Lecturers) (New-Query)
{Lecturer information} Anything else?
U5: (Question:Registration) [0.8]
Degree: Computer Science[0.9]
The registration information in Computer Science
S6: (Answer:Registration) (New-Query)
{Registration information} Anything else?
U6: Negation [0.8]
S7: (Closing)
Thank you for using the UAH system.

Fig. 5. An example of a dialog acquired by means of the user simulation technique (translation from Spanish to English).

After the welcome prompt of the system, the user simulator asks for information about lecturers. The error simulator assigns a low confidence to what it has understood (0.1), so its selects to confirm this query in S2. Then, the user simulator confirms the query about lecturers and a high confidence value is assigned to this response in turn U2. Next, the system asks for the name of the lecturer (S3), which is provided by the user simulator in turn U3. Again, a low confidence value is assigned to this information, and the name of the lecturer is confirmed in turn S4. Once the user simulator has confirmed that the provided value for the name of the subject is valid (turn U4), the system provides the specific information in turn S5. In the following turn the user simulator asks for registration deadlines for a specific degree (turn U5). Due to the error simulator assigns a high confidence value to this information, the system directly provides an answer to this query in turn S6. Once the user simulator indicates that no additional information is required to fulfill the objectives of this dialog, the system generates a closing prompt (S7).

Finally, we evaluated the behavior of UAH with recruited users using the same set of scenarios designed for the user simulation. A total of 600 dialogs were recorded from the interactions of 150 users, 75 users employed CA_1 and 75 users employed CA_2. Additionally, these users were asked to fill in a questionnaire with their opinion about several aspects of the interaction. The results of the objective evaluation using the corpus and the subjective evaluation using the information provided in the questionnaires will be discussed in Section 6.4.

6. Discussion of results

We have considered two dimensions in order to evaluate the initial conversational agent for the UAH task and its evolution once the simulated dialogs are included to learn a new dialog model for the conversational agent: high-level features (dialog success and duration) and dialog style features (speech-act frequency and proportion of goal-directed actions). Using these measures, we tried to evaluate the success of the simulated dialogs as well as its efficiency and variability regarding the different functionalities of the system.

By means of high-level measures in our evaluation, we focus on the success rate and efficiency of the system. We are particularly interested in goal achievement rates and goal completion times. Only the dialogs were considered successful if they fulfill the complete list of objectives that has been previously defined in the corresponding scenario.

Six high-level dialog features have been defined for the evaluation of the dialogs: the average duration of the dialogs in terms of number of turns per dialog, the percentage of different dialogs without considering the attribute values, the number of turns of the most seen dialog, the number of turns of the shortest dialog, and the number of turns of the longest dialog, the percentage of unseen situations (i.e., the dialog situations in terms of dialog states that are present in the test partition but are not present in the corpus used for learning the conversational agent), and the percentage of answers provided by the conversational agent that would cause the failure of the dialog.

For dialog style features, we define and count a set of system/user dialog acts. On the system side, we have measured the confirmation of concepts and attributes, questions to require information, and system answers generated after a database query. On the user side, we have measured the percentage of turns in which the

user carries out a request to the system, provides information, confirms a concept or attribute, the Yes/No answers, and other answers not included in the previous categories. Finally, we have measured the proportion of goal-directed actions (request and provide information) versus the grounding actions (confirmations) and rest of actions.

6.1. High-level dialog features

Table 2 shows the comparison of the different high-level measures defined for the evaluation. It can be observed that when CA_2 was used, there was a reduction in the average number of turns and in the number of turns in the longest, shortest and most observed dialogs. These results show that improving the dialog strategy made it possible to reduce the number of necessary system actions to attain the scenario goals.

In addition, the results show a higher variability in the dialogs generated with CA_2 as there was a higher percentage of different dialogs and the most observed dialog was less repeated. There was also a slight increment in the mean values of the turn length for the dialogs collected with CA_2. These dialogs had 1.7 actions per user turn instead of the 1.3 actions observed for CA_1, due to the better selection of the system actions in the improved strategy. Regarding participant activity, the dialogs generated with CA_2 had a higher proportion of system actions because they required less confirmations.

The shape of the distributions for the task length of the dialogs (Fig. 6) shows that the CA_1 dialogs have the largest standard deviation given that the task length of these dialogs is more disperse. CA_2 dialogs have minimum deviation since the successful dialogs are usually those which require the minimum number of turns to achieve the objective(s) predefined for the different scenarios.

6.2. Evolution of the dialog strategy

Figure 7 shows the previously described evolution of the average duration in terms of total dialog turns (*duration*). It also shows the reduction in the number of responses provided by the conversational agent which cause a failure of the dialog (*#error*) and the number of unseen situations for which there is not a system response in the dialog model (*#unseen*). As it can be seen from these results, the CA_2 system not only reduces the time required to fulfill each one of the objectives of the dialog, but also it reduces the possibility of selecting an erroneous response.

Table 2
High-level dialog measures obtained for DM_1 and DM_2

High-level groups	Dialog features	CA_1	CA_2
Dialog length	Average number of turns per dialog	12.9 ± 2.3	9.8 ± 1.6
Different dialogs	Percentage of different dialogs (%)	62.9	78.3
	Repetitions of the most observed dialog	15	3
Turn length	Average number of actions per turn	1.3	1.7
Participant activity	Number of user turns of the most observed dialog	9	7
	Number of user turns of the shortest dialog	7	5
	Number of user turns of the longest dialog	13	9

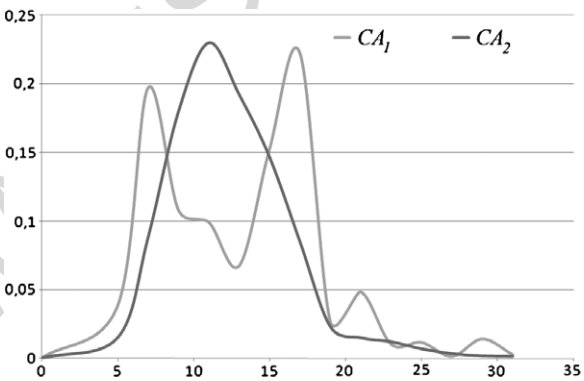

Fig. 6. Task length distribution.

6.3. Dialog style and cooperativeness

We were also interested in the frequency of user and system dialog acts, as well as in the proportion of goal-directed dialog acts to request or provide information, versus the grounding dialog acts to confirm data. On the system side, *S_request*, *S_confirm*, and *S_inform* indicate actions through which the system respectively requests, confirms, or provides information. *S_other* stands for other types of system prompts (e.g, *Waiting* and *Not-Understood* dialog acts). On the user side, *U_provide*, *U_query*, *U_confirm*, and *U_yesno* respectively identify actions by which the user provides, requests, confirms information or gives a yes/no answer, while *U_unknown* represents all other user actions (e.g., dialog formalities or out of task information). The experimental results are set out in Table 3.

It can be observed that there are significant differences for both dialog models. On the one hand, users

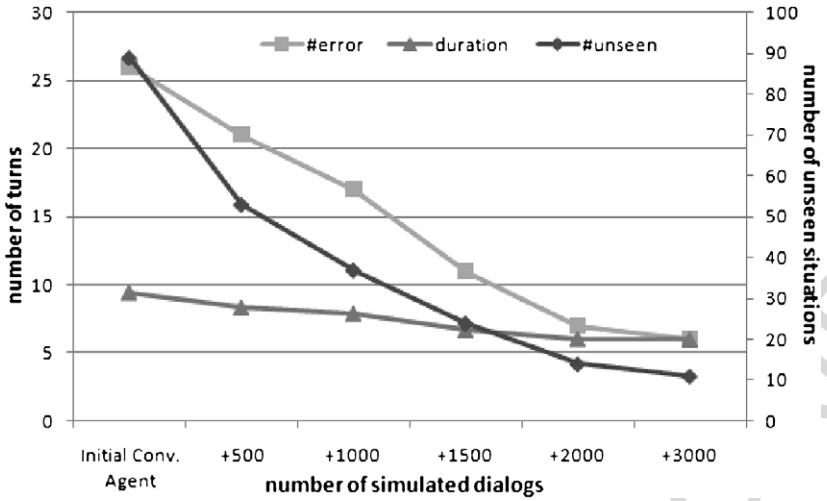

Fig. 7. Evolution of the number of unseen situations, number of errors, and average number of turns.

Table 3
Percentages of user and system dialog acts using CA_1 and CA_2

	CA_1	CA_2
U_Request	31.74	35.92
U_Provide	21.72	25.43
U_Confirm	10.81	7.89
U_Yes/No	34.47	30.21
U_Unknown	1.26	0.55
S_Confirm	17.41	12.76
S_Request	18.32	17.23
S_Inform	63.80	69.79
S_Other	0.47	0.22

needed to employ less confirmation turns with CA_2, which explains the higher proportion of dialog acts to request and provide information. On the other hand, using CA_2 there was an increment in the number of system turns providing information to the user, which is consistent with the fact that the task completion rate is higher using this DM.

In addition, we grouped all user and system actions into three categories: "goal directed" (actions to provide or request information), "grounding" (confirmations and negations), and "other". The results show that the dialogs acquired with CA_2 are better as the proportion of goal-directed actions increases (from 63.46% using CA_1 to 75.12% using CA_2).

6.4. Evaluation with real users

Finally, we have evaluated the behavior of the two conversational agents with real users. A total of 600 dialogs was recorded from the interactions of 150 students and lecturers in our department employing the initial and final conversational agents. An objective and subjective evaluation were carried out. We considered the following measures for the objective evaluation:

(1) Dialog success rate. This is the percentage of successfully completed tasks. In each scenario, the user has to obtain one or several items of information, and the dialog success depends on whether the system provides correct data (according to the aims of the scenario) or incorrect data to the user.
(2) Average number of turns per dialog (nT).
(3) Confirmation rate. It was computed as the ratio between the number of explicit confirmations turns (nCT) and the number of turns in the dialog (nCT/nT).
(4) Average number of corrected errors per dialog (nCE). The average of errors detected and corrected by the dialog manager. We have considered only those which modify the values of the attributes and thus could cause the failure of the dialog. The errors are detected using the confidence scores provided by the ASR and NLU modules. Implicit and explicit confirmations are employed to confirm or require again values detected with low reliability.
(5) Average number of uncorrected errors per dialog (nNCE). This is the average of errors not corrected by the dialog manager. Again, only errors that modify the values of the attributes are considered.

Table 4
Results of the objective evaluation with real users

	Success	nT	Confirmation	ECR
CA_1	86.0%	14.4	34%	0.87%
CA_2	93.0%	11.2	26%	0.98%

Table 5
Results of the subjective evaluation with recruited users (1 = lowest, 5 = highest)

	Q1	Q2	Q3	Q4	Q5	Q6
CA_1	4.6	3.6	3.8	3.4	3.2	4.1
CA_2	4.7	3.9	4.3	4.2	3.3	4.6

(6) Error correction rate (%ECR). The percentage of corrected errors, computed as nCE/(nCE + nNCE).

The results, presented in Table 4, show that the success rate is above 86% for both dialog models, and that CA_2 improved it by 7% absolute. Using CA_2, the average number of turn per dialog was reduced from 14.4 to 11.2. These values are slightly higher for both systems than those attained for the simulated users (set out in Table 2). The reason is that in some dialogs the recruited users either provided information that was not mandatory or asked for information that was not specified in the scenarios.

The confirmation and error correction rates were also improved by using CA_2 as it required less data from the user, thus reducing the number of ASR errors. A problem occurred when the user input was misrecognized but it had high confidence score, in which case it was forwarded to the conversational agent. However, as the success rate shows, this problem did not have a remarkable impact on system performance.

Secondly, we carried out a subjective evaluation by asking the recruited users to fill in a questionnaire comprised of the following questions: (i) Q1: *How well did the system understand you?*; (ii) Q2: *How well did you understand the system?*; (iii) Q3: *Was it easy for you to obtain the requested information?*; (iv) Q4: *Was the interaction with the system quick enough?*; (v) Q5: *Was it easy to correct system errors?*; (vi) Q6: *In general, are you satisfied with the performance of the system?* The possible answers for each one of the questions were the same: *Never/Not at all*, *Seldom/In some measure*, *Sometimes/Acceptably*, *Usually/Well*, and *Always/Very Well*. All the answers were assigned a numeric value between one and five (in the same order as they appear in the questionnaire). Table 5 shows the average results obtained.

It can be observed that using either CA_1 or CA_2 the users perceived that the system understood them correctly. Moreover, they expressed a similar opinion regarding the easiness for correcting system errors. However, users said that it was easier to obtain the information specified in the scenario using CA_2, and that the interaction with the system was more adequate with this dialog manager. Finally, the users were more satisfied with the system employing CA_2.

7. Conclusions

In this paper, we have described a technique for simulating user agents and evaluate conversational agents. Our technique is based on a statistical model which takes the complete history of the interaction into account to decide the next user answer. This decision is modeled by a classification process in which a neural network is used. An additional statistical model has been introduced for errors introduction and confidence measures generation. This way, the conversational agent can also be evaluated by considering different conditions in the communication channel.

Our statistical user simulator provides the user intention level in terms of the semantic representation that would be generated by the ASR and NLU modules in the architecture of a conversational agent. This way, dialogs are automatically labeled during the simulation using the semantics defined for the task. Thus, the proposed user simulation methodology allows the generation of new dialogs with little effort and the adaptation to a new task is also simplified.

We have described the application of our proposal to develop an enhanced academic conversational agent for the UAH task. Different measures have been defined to evaluate high-level dialog features, dialog style features, and different statistics of the acquired dialog corpora. Using these measures, we evaluate the success of the dialogs as well as their efficiency and variability with regard to the different objectives specified in the task.

In our proposal, the simulated dialogs are used to automatically reinforce the dialog model of the conversational agent. The results of the evaluation of the conversational agents developed for the UAH task show that the proposed user simulation methodology can be used not only to evaluate conversational agents but also to explore new enhanced dialog strategies. Carrying out these tasks with a non-automatic approach would

require a very high cost that sometimes is not affordable.

By means of the simulated dialogs, the conversational agent reduces the time needed to complete the dialogs, thereby allowing the conversational agent to tackle new dialog situations and generate new coherent answers for the situations already present in an initial dialog model. This way, the conversational agent reduces the number of system turns for the different kinds of dialogs, and automatically detect different valid paths to achieve each of the required objectives defined for the task.

As a future work, we are adapting the proposed user simulation technique for its application in more difficult domains in our multi-agent architecture. We also want to evaluate the complete MAS architecture with real users and systems offering similar functionalities, extending the described evaluation to measure the influence of the rest of agents in the operation of the developed conversational agent. Finally, we also want to evaluate the influence of taking into account different context information sources to improve the operation of a multiagent system developed following our proposed architecture.

Acknowledgements

This work was supported in part by Projects MINECO TEC2012-37832-C02-01, CICYT TEC 2011-28626-C02-02, CAM CONTEXTS (S2009/TIC-1485).

References

[1] H. Ai, A. Raux, D. Bohus, M. Eskenazi and D. Litman, Comparing spoken dialog corpora collected with recruited subjects versus real users, in: *Proc. of the 8th SIGdial Workshop on Discourse and Dialogue*, Antwerp, Belgium, 2007, pp. 124–131.

[2] A. Ballinas, A. Munoz and A. Rangel, Multiagent system applied to the modeling and simulation of pedestrian traffic in counterflow, *Journal of Artificial Societies and Social Simulation* **14** (2011), 25–38.

[3] M. Balmer and K. Nagel, Shape morphing of intersection layouts using curb side oriented driver simulation, in: *Innovations in Design and Decision Support Systems in Architecture and Urban Planning*, Springer-Verlag, 2006, pp. 167–183.

[4] S. Bandini, F. Celada, S. Manzoni, R. Puzone and G. Vizzari, Modelling the immune system with situated agents, *Lecture Notes in Computer Science* **3931** (2006), 231–243.

[5] S. Bandini, S. Manzoni and G. Vizzari, Agent based modeling and simulation: An informatics perspective, *Journal of Artificial Societies and Social Simulation* **12** (2009), 97–126.

[6] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford Univ. Press, 1995.

[7] D. Bohus, S. Grau, D. Huggins-Daines, V. Keri, G. Krishna, R. Kumar, A. Raux and S. Tomko, Conquest – An open-source dialog system for conferences, in: *Proc. of the 7th Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL'07)*, Rochester, NY, USA, 2007, pp. 9–12.

[8] D. Bohus and A. Rudnicky, LARRI: A language-based maintenance and repair assistant, in: *Proc. of the Multi-Modal Dialogue in Mobile Environments Conference (IDS'02)*, Kloster Irsee, Germany, 2002, pp. 203–218.

[9] J. Bos, E. Klein, O. Lemon and T. Oka, DIPPER: Description and formalisation of an information-state update dialogue system architecture, in: *Proc. of the 4th SIGdial Workshop on Discourse and Dialogue*, Sapporo, Japan, 2003, pp. 115–124.

[10] S. Brahnam, Building character for artificial conversational agents: Ethos, ethics, believability and credibility, *Psychnology Journal* **7** (2009), 9–47.

[11] Z. Callejas and R. López-Cózar, Relations between de-facto criteria in the evaluation of a spoken dialogue system, *Speech Communication* **50** (2008), 646–665.

[12] G. Chung, Developing a flexible spoken dialog system using simulation, in: *Proc. of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, Barcelona, Spain, 2004, pp. 63–70.

[13] H. Cuayáhuitl, S. Renals, O. Lemon and H. Shimodaira, Human-computer dialogue simulation using hidden Markov models, in: *Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU'05)*, San Juan, Puerto Rico, 2005, pp. 290–295.

[14] J.F. de Alba and J. Pavón, Talking agents: A distributed architecture for interactive installation-art works, *Integrated Computer-Aided Engineering* **17** (2010), 243–259.

[15] W. Eckert, E. Levin and R. Pieraccini, User modeling for spoken dialogue system evaluation, in: *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU'97)*, Santa Barbara, USA, 1997, pp. 80–87.

[16] W. Eckert, E. Levin and R. Pieraccini, Automatic evaluation of spoken dialogue systems, Technical Report TR98.9.1, ATT Labs Research, 1998.

[17] S. Espana, F. Zamora, M. Castro and J. Gorbe, Efficient BP algorithms for general feedforward neural networks, *Lecture Notes in Computer Science* **4527** (2007), 327–336.

[18] R. Fikes and T. Kehler, The role of frame-based representation in knowledge representation and reasoning, *Communications of the ACM* **28** (1985), 904–920.

[19] P. Filipe and N. Mamede, Ambient intelligence interaction via dialogue systems, in: *Ambient Intelligence*, In-teh, 2010, pp. 109–124.

[20] F. García, L. Hurtado, E. Sanchis and E. Segarra, The incorporation of confidence measures to language understanding, *Lecture Notes in Computer Science* **2807** (2003), 165–172.

[21] K. Georgila, J. Henderson and O. Lemon, Learning user simulations for information state update dialogue systems, in: *Proc. of the 9th European Conference on Speech Communication and Technology (Eurospeech'05)*, Lisbon, Portugal, 2005, pp. 893–896.

[22] F.E. Gérard Bailly and S. Raidt, Gaze, conversational agents and face-to-face communication, *Speech Communication* **52** (2010), 598–612.

[23] J. Glass, G. Flammia, D. Goodine, M. Phillips, J. Polifroni, S. Sakai, S. Seneff and V. Zue, Multilingual spoken-language understanding in the MIT Voyager system, *Speech Communication* **17** (1995), 1–18.

[24] D. Griol, Z. Callejas and R. López-Cózar, Statistical dialog management methodologies for real applications, in: *Proc. of the 11th SIGdial Meeting*, 2010, pp. 124–131.

[25] D. Griol, L. Hurtado, E. Segarra and E. Sanchis, A statistical approach to spoken dialog systems design and evaluation, *Speech Communication* **50** (2008), 666–682.

[26] D. Griol, G. Riccardi and E. Sanchis, Learning the structure of human-computer and human-human dialogs, in: *Proc. of the 10th Interspeech Conference*, 2009, pp. 2775–2778.

[27] D. Griol, N. Sánchez-Pi, J. Carbó and J. Molina, An architecture to provide context-aware services by means of conversational agents, *Advances in Intelligent and Soft Computing* **79** (2010), 275–282.

[28] B. Heath, R. Hill and F. Ciarallo, A survey of agent-based modeling practices (January 1998–July 2008), *Journal of Artificial Societies and Social Simulation* **12** (2009), 1–9.

[29] S. Jung, C. Lee, K. Kim, D. Lee and G. Lee, Hybrid user intention modeling to diversify dialog simulations, *Computer Speech and Language* **25** (2011), 307–326.

[30] F. Klügl, R. Herrler and C. Oechslein, From simulated to real environments: How to use sesam for software development, *Lecture Notes in Computer Science* **2831** (2003), 13–24.

[31] E. Levin, R. Pieraccini and W. Eckert, A stochastic model of human–machine interaction for learning dialog strategies, *IEEE Transactions on Speech and Audio Processing* **8** (2000), 11–23.

[32] B. Lin and L. Lee, Computer aided analysis and design for spoken dialogue systems based on quantitative simulations, *IEEE Trans. Speech Audio Process* **9** (2001), 534–548.

[33] D.J. Litman and S. Silliman, ITSPOKE: An intelligent tutoring spoken dialogue system, in: *Proc. of the 4th HLT/NAACL Conference*, 2004, pp. 233–236.

[34] R. López-Cózar and M. Araki, *Spoken, Multilingual and Multimodal Dialogue Systems*, Wiley, 2005.

[35] R. López-Cózar, A. D. la Torre, J. Segura, A. Rubio and V. Sánchez, Assessment of dialogue systems by means of a new simulation technique, *Speech Communication* **40** (2003), 387–407.

[36] C. Macal and M. North, Tutorial on agent-based modelling and simulation, *Journal of Simulation* **4** (2010), 151–162.

[37] M.F. McTear, *Spoken Dialogue Technology: Towards the Conversational User Interface*, Springer, 2004.

[38] H. Melin, A. Sandell and M. Ihse, CTT-bank: A speech controlled telephone banking system – An initial evaluation, in: *TMH Quarterly Progress and Status Report (TMH-QPSR)*, Vol. 1, 2001, pp. 1–27.

[39] P. Menezes, F. Lerasle, J. Dias and T. Germa, Towards an interactive humanoid companion with visual tracking modalities, in: *Humanoid Robots, Human-like Machines*, Advanced Robotic Systems Int. and I-Tech Education and Publishing, 2007 pp. 48–78.

[40] M. Minsky, A framework for representing knowledge, in: *The Psychology of Computer Vision*, McGraw-Hill, 1975, pp. 211–277.

[41] S. Möller, R. Englert, K. Engelbrecht, V. Hafner, A. Jameson, A. Oulasvirta, A. Raake and N. Reithinger, MeMo: towards automatic usability evaluation of spoken dialogue services by user error simulations, in: *Proc. of the 9th Int. Conference on Spoken Language Processing (Interspeech/ICSLP)*, Pittsburgh, PA, USA, 2006, pp. 1786–1789.

[42] L. Navarro, F. Flacher and V. Corruble, Dynamic level of detail for large scale agent-based urban simulations, in: *Proc. of the 10th Int. Conference on Autonomous Agents and Multiagent Systems (AAMAS'11)*, Taipei, Taiwan, 2011, pp. 701–708.

[43] M. North, N. Collier and J. Vos, Experiences creating three implementations of the repast agent modeling toolkit, *ACM Transactions on Modeling and Computer Simulation* **16** (2006), 1–25.

[44] T. Paek and R. Pieraccini, Automating spoken dialogue management design using machine learning: An industry perspective, *Speech Communication* **50** (2008), 716–729.

[45] J. Pavón, C. Sansores, J. Gómez and F. Wang, Modelling and simulation of social systems with INGENIAS, *Int. Journal of Agent-Oriented Software Engineering* **2** (2008), 196–221.

[46] R. Pieraccini, *The Voice in the Machine: Building Computers that Understand Speech*, The MIT Press, 2012.

[47] O. Pietquin and T. Dutoit, A probabilistic framework for dialog simulation and optimal strategy learning, *IEEE Transactions on Speech and Audio Processing* **14** (2005), 589–599, Special Issue: Data Mining of Speech, Audio and Dialog.

[48] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning internal representations by error propagation, in: *PDP: Computational Models of Cognition and Perception, I*, MIT Press, 1986, pp. 319–362.

[49] N. Sánchez-Pi, V. Fuentes, J. Carbó and J. Molina, Knowledge-based system to define context in commercial applications, in: *Proc. of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD'07)*, 2007, pp. 694–699.

[50] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye and S. Young, Agenda-based user simulation for bootstrapping a POMDP dialogue system, in: *Proc. of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*, Rochester, NY, USA, 2007, pp. 149–152.

[51] J. Schatzmann, B. Thomson, K. Weilhammer, H. Ye and S. Young, Agenda-based user simulation for bootstrapping a POMDP dialogue system, in: *Proc. of Human Language Technologies HLT/NAACL'07 Conference*, Rochester, NY, USA, 2007, pp. 149–152.

[52] J. Schatzmann, B. Thomson and S. Young, Statistical user simulation with a hidden agenda, in: *Proc. of the 8th SIGdial Workshop on Discourse and Dialogue*, Antwerp, Belgium, 2007, pp. 273–282.

[53] J. Schatzmann, K. Weilhammer, M. Stuttle and S. Young, A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies, *Knowledge Engineering Review* **21** (2006), 97–126.

[54] K. Scheffler and S. Young, Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning, in: *Proc. of Human Language Technology (HLT'02)*, San Diego, CA, USA, 2001, pp. 12–18.

[55] K. Scheffler and S. Young, Corpus-based dialogue simulation for automatic strategy learning and evaluation, in: *Proc. of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001). Workshop on Adaptation in Dialogue Systems*, Pittsburgh, PA, USA, 2001, pp. 64–70.

[56] C. Sharma and J. Kunins, *VoiceXML: Strategies and Techniques for Effective Voice Application Development with VoiceXML 2.0*, Wiley, 2001.

[57] F. Torres, E. Sanchis and E. Segarra, User simulation in a stochastic dialog system, *Computer, Speech and Language* **22** (2008), 230–255.

[58] C. Vaquero, O. Saz, E. Lleida, J. Marcos and C. Canalís, VOCALIZA: An application for computer-aided speech therapy in Spanish language, in: *Proc. of the IV Jornadas en Tecnología del Habla*, Zaragoza, Spain, 2006, pp. 321–326.

[59] F. Weng, S. Varges, B. Raghunathan, F. Ratiu, H. Pon-Barry, B. Lathrop, Q. Zhang, T. Scheideck, H. Bratt, K. Xu, M. Purver, R. Mishra, M. Raya, S. Peters, Y. Meng, L. Cavedon and L. Shriberg, CHAT: A conversational helper for automotive tasks, in: *Proc. of the 9th Int. Conference on Spoken Language Processing (Interspeech/ICSLP)*, Pittsburgh, PA, USA, 2006, pp. 1061–1064.

[60] D. Weyns, N. Boucké and T. Holvoet, Gradient field-based task assignment in an AGV transportation system, in: *Proc. of the 5th Int. Conference on Autonomous Agents and Multiagent Systems (AAMAS'06)*, Hakodate, Japan, 2006, pp. 842–849.

[61] U. Wilensky and W. Rand, *An Introduction to Agent-Based Modeling: Modeling Natural, Social and Engineered Complex Systems with NetLogo*, MIT Press, 2009.

[62] J. Williams and S. Young, Partially observable Markov decision processes for spoken dialog systems, *Computer Speech and Language* **21** (2007), 393–422.

[63] P. Windrum, G. Fagiolo and A. Moneta, Empirical validation of agent-based models: Alternatives and prospects, *Journal of Artificial Societies and Social Simulation* **10** (2007), 1–8.

[64] S. Young, The statistical approach to the design of spoken dialogue systems, Technical report, CUED/F-INFENG/TR.433, Cambridge University Engineering Department, Cambridge, UK, 2002.

[65] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen and L. Hetherington, JUPITER: A telephone-based conversational interface for weather information, *IEEE Transactions on Speech and Audio Processing* **8** (2000), 85–96.